To the Comal80 Standardisation Group

Proposal

In order to complete the Comal80 Kernel the three Danish manufacturers, Dansk Data Elektronik, Regnecentralen, and Unicomal, have met a couple of times in january and february 1985. The following proposal is for the present the result of these meetings, and agreed by all three manufacturers. The proposal suggests minor adjustments to the standard, but most of the proposal should be regarded as clarifications.

The production numbers and the page numbers refer to the Comal Kernel, April 13-14, 1984, with corrections of september 1984.

1. The procedure call statement is allowed with or without parentheses surrounding the parameter list. Production 134 is changed to

   134. <procedure call statement> ::=
   [EXEC] <procedure identifier> [<parameter list>]

   <parameter list> ::= <parenthesized parameter list> |
   <actual parameter list>

   <parenthesized parameter list> ::=
   ( <actual parameter list> )

2. A randomize statement is introduced:

   <randomize statement> ::=
   RANDOMIZE [<numeric expression>]

   Prior to the execution of a program the random generator is seeded by a random value.

The execution of a randomize statement without a numeric expression means that the random generator is seeded by a random value.
The execution of a randomize statement with a numeric expression means that the random generator is seeded by the value of the numeric expression.

3. RND is allowed with zero or two arguments only. Page 28 rule 11 is changed to

   11. RND
   --- RND returns a "random" value greater than zero and less than one.

4. The open statement is changed to

   114. <open statement> ::= OPEN FILE <channel number>,
                              <file name>, <mode>

   Production 115 is ruled out.

   117. <mode> ::= READ ¦ WRITE ¦ APPEND ¦
                   RANDOM [<random mode>] <record length>

   <random mode> ::= READONLY ¦ WRITEONLY

   The semantics following production 118 is changed to:

   If mode is random all records must be of the length specified. The string expression should evaluate to a file name.

   If the file does or does not exist the system will respond according to the following plan:

| mode | file exists: yes | no |
|---|---|---|
| READ | open | runtime error |
| WRITE | runtime error | create |
| APPEND | open | create |
| RANDOM | open | create |
| RANDOM READONLY | open | runtime error |
| RANDOM WRITEONLY | open | create |

5. Production 121 is changed to

   121. <delete statement> ::= DELETE <file name>

6. The first paragraph of the semantics following production 46 is changed to:

   Functions and procedures differ only in the way thay are invoked and in the way they return values. A procedure is called by a procedure call, whereas a function is called by the use of the function identifier in an expression. The effect of a function invokation is to provide a single returned value of the same type as the function identifier. Procedures can not return a value so procedure identifiers have no type (see also the RETURN statement).

7. The second paragraph of the semantics following production 46 is changed to:

   The procedure or function can only be entered by calling it, and can only be exited by executing a RETURN statement or by control reaching the end of the procedure. The procedure or function identifier in the ENDPROC or ENDFUNC must match that in the heading. Parameters, other than arrays, are passed by value unless REF (pass by reference) is specified. Arrays are always passed by

reference so REF must be included for them. Their number of dimensions are indicated by commas.

and the semantics following production 96 is changed to:

Return causes the current procedure or function to be exited. A function must be exited through a RETURN with an expression. A return with an expression is not allowed within a procedure. A return of any sort may not be used within the main program.

8. To the semantics following production 109 add:

A label statement is also used to separate DATA statements (see RESTORE statement).

9. Production 8 is changed to:

8. <unstructured statement> ::=
<simple statement> <eol> |
<label statement> |
<eol>

thus allowing empty statements.

10. The examples at the bottom of page 15 is changed to (in order to make it possible to enter the examples):

```
  36  MOD 5 is 1          36  MOD (-5) is undefined
(-36) MOD 5 is 4        (-36) MOD (-5) is undefined
  35) MOD 5 is 0          35  MOD (-5) is undefined
(-35) MOD 5 is 0        (-35) MOD (-5) is undefined

  36  DIV 5 is  7         36  DIV (-5) is -8
(-36) DIV 5 is -8       (-36) DIV (-5) is  7
  35  DIV 5 is  7         35  DIV (-5) is -7
(-35) DIV 5 is -7       (-35) DIV (-5) is  7
```

11. In order to obtain consistency production 44 is changed to:

    44. <import statement> ::= IMPORT <import identifier>
                                {,<import identifier>}

        <import identifier> ::= <variable identifier> |
                                <procedure identifier> |
                                <function identifier>

    and the binding rules at page 26 are changed to:

    BINDING:

    1. Procedure and function must be IMPORTed into CLOSED procedures or functions.
    2. OPEN procedures or functions can not be imported.

12. The implicit rounding function is defined as:

    implicit_rounding(x) = int(x+0.5)

    As an extension to the kernel it is further suggested to add three new built-in functions

    round(x)
    roundeven(x)
    float(x)

    which are hoped to be selfexplanatory.

13. To make it possible to define assignment of real expressions to integer variables either at a later stage or in the actual implementations, the semantics at 15.1 (page 30) is changed to:

    Some flexibility between reals and integers is allowed. Integer numeric expressions may be assigned to real

variables without any errors. If a real expression is assigned to an integer variable, the result is dependent on the actual implementation.

14. Add the following semantics in front of the semantics following production 129:

    The printline is divided into print zones. The zone width is determined by the zone statement. The default zone width is 0. An odd sized zone may occur at the end of the print line.
    Printing zone divided lines may be compared to using the tabulator key on an ordinary typewriter except that the "tabulator key" has no effect when the print position is at the beginning of a zone. Using a zone width of 0 or 1 has a visible effect when printing the empty string only.
    If <print separator> is comma the ouput of the next element starts from the leftmost position of the first free zone. If there are no more free zone on the current line, printing continues on the next line.
    If <print separator> is semicolon the output of the next print element starts from the next print position. A numeric value is printed with a trailing space.
    The effect of a <print end> is the same as a <print separator>. If no <print end> is specified, printing is continued in the first position on the next line.
    A print statement with no <print list> causes output of a new line.

15. In order to distinguish between the value of the IN operator in the cases

    ```
    ""  IN "abc"      and
    "a" IN "abc"
    ```

    we recommend that

"" IN "abc" is 4, that is len("abc") + 1.

Thus, add to the semantics following production 65:

If <string1> is evaluated to the empty string the value is the length of <string2> + 1.

16. Change production 101 to

```
<input statement> ::= INPUT [<string constant>:]
                        <variable> [<print end>] |
                      INPUT FILE <file designator> :
                        <variable list>
```

Further, the standard prompt should be left implementation dependent.

Change the semantics after production 106 to:

The <string constant> is used as a prompt. If the prompt is absent then a system standard prompt is supplied. The standard prompt is implementation dependent.
The system waits for the user to input a value. If the user makes a mistake and types in a value that does not match the type of the variable, the system prints an error message and re-issues the prompt.
If <print end> is not specified the following print position will be the first position on the next line. If the <print end> is specified the rules from the print statement apply.

17. In appendex C the words EXITIF, LET, and USE should be excluded and according to the suggestions in this paper READONLY, WRITEONLY, and RANDOMIZE should be added.

Anders Ansted, Dansk Data Elektronik
Jens Erik Jensen, Unicomal
Erik Jeppesen, Regnecentralen
Birgit Landgrebe, Regnecentralen
Helge Lassen, Unicomal